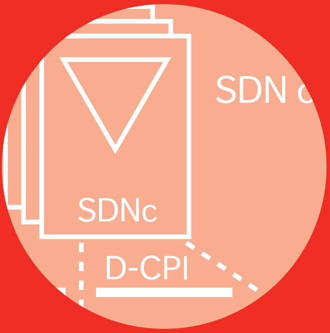
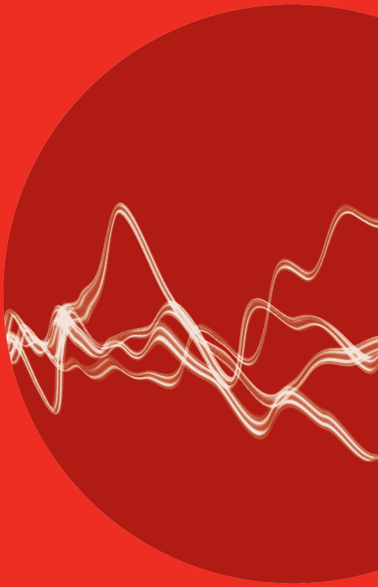
CHARTING THE FUTURE OF INNOVATION VOLUME 92 | #7.2015

ERICSSON TECHNOLOGY

# Review





# IDENTIFYING AND ADDRESSING THE VULNERABILITIES AND SECURITY ISSUES OF SDN

ERICSSON

# IDENTIFYING AND **ADDRESSING** THE vulnerabilities & **SECURITY ISSUES** OF SDN

The promises of agility, simplified control, and real-time programmability offered by software-defined networking (SDN) are attractive incentives for operators to keep network evolution apace with advances in virtualization technologies. But do these capabilities undermine security? To answer this question, we have investigated the potential vulnerabilities of SDN. The aim is for this architecture to serve as a secure complement to cloud computing, and to ensure that networks are protected from attack by malicious intruders.

KRISTIAN SLAVOV
DANIEL MIGAULT
MAKAN POURZANDI

**TRADITIONAL NETWORK architecture has reached the point where its ability to adapt to dynamic environments, like those enabled by virtualization technologies, has become a hindrance. By separating the control plane from the data plane, SDN raises the level of system abstraction, which in turn opens the door for network programmability, increased speed of operations, and simplification: in short, the key to delivering on its promises, and enabling telecom networks and IT to develop in parallel.**

At the heart of SDN architecture lies the SDN controller (SDNC). Logically positioned between network elements (NEs) and SDN applications (SDN apps), the SDNC provides an interface between the two. Its centralized position enables it to provide other SDN components with a global overview of what is happening in the network; it can configure NEs on the fly and determine the best path for traffic. The SDNC and the shift to centralized control set SDN architecture apart from traditional networks – in which control is distributed. Unfortunately, the centralized position of the SDNC makes it a primary surface for attack.

For the purposes of this article, we limited the scope of our study into the vulnerabilities of SDN to the single controller use case (with one controller governing the data plane), even though SDN architecture allows for several. Our discussion covers the SDN elements and their interactions in the single controller case, as well as the interactions between the SDNC and the management plane.

### Why centralize?

As defined by ONF[1], a logically centralized control plane makes it possible to maintain a network-wide view of resources, which can then be exposed to the application layer. To provide such a centralized architecture, SDN uses one or more NEs that interface with the SDNC. The benefit of building networks in this way is simplified network management, and improved agility.

Centralization equips networks for programmability, which in turn increases autonomy. One possibility enabled by programmability is the automatic detection and mitigation of DDOS attacks, which results in rapid resolution of any problems that may arise. Programmability also allows network resources to be shared automatically, which – together with the capability to create virtual networks created on top of existing network infrastructure – enables automatic sharing by multiple tenants.

### Benefits and vulnerabilities

SDN facilitates the integration of security appliances into networks, which can be implemented directly on top of the control plane, rather than being added as separate appliances or instantiated within multiple NES. SDN's centralized management approach enables events within the entire network to be collected and aggregated, The resulting broader, more coherent and more accurate image of the network's status, makes security strategies both easier to enforce and to monitor.

The ability to implement security mechanisms directly on top of the controller or on steering traffic at run time (using legacy appliances when necessary) makes it possible to dynamically add taps and sensors at various places in the network – which makes for more effective network monitoring. With an accurate picture of its status, the network can more readily detect attacks, and the number of false positives reported can be reduced. In practice, if a tap indicates to the SDNC that a device is showing signs of being hijacked by a botnet, the SDNC can steer the potentially offending traffic to an IDS for analysis and monitoring. If the traffic is deemed malicious by the IDS, the SDNC can filter it and instruct the first-hop NE accordingly.

Its ability to facilitate the collection of network-status information as well as enabling automatic detection and resolution of any breach in security, makes SDN ideal for integration into network threat intelligence centers and Service Operation Centers (SOCS). Unfortunately, the rich feature set of SDN also provides a larger attack surface compared with traditional networks – an issue documented in a number of recently published research papers[2].

### Reference model

The overall SDN architecture comprises the following elements:

- **NES – which are responsible for forwarding packets to the next appropriate NE or end host;**
- **SDNC – which sends forwarding rules on to the NES according to instructions it receives from SDN apps;**

### Terms and abbreviations

**DDOS**–Distributed DoS | **DOS**–Denial of Service | **GRE**–Generic Routing Encapsulation | **IDS**–intrusion detection system | **IPSEC**–Internet Protocol Security | **MM**– management module | **MPLS**–multi-protocol label switching | **NE**–network element | **ONF**–Open Networking Foundation | **RBAC** role-based access control | SDNsoftware-defined networking | **SDNC**–SDN controller | **SLA**–Service Level Agreement | **TLS**–Transport Layer Security domain-specific modeling language

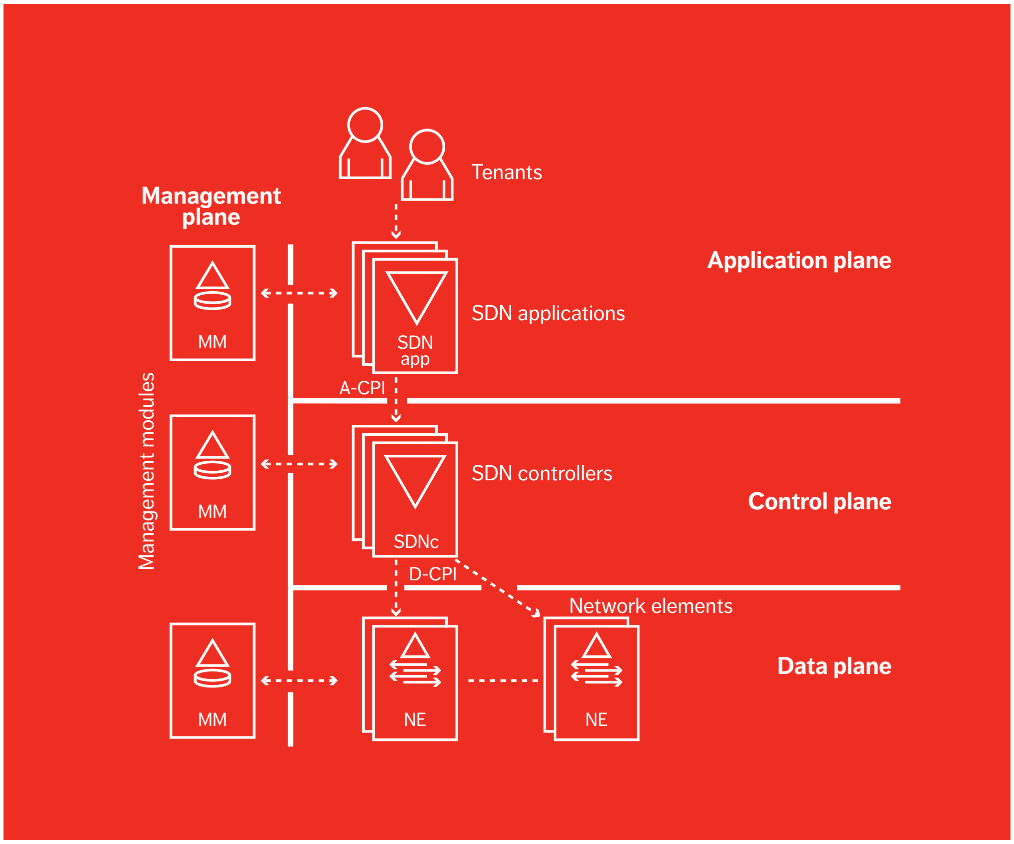


*Figure 1*
SDN architecture

- **SDN apps – which issue commands to dynamically configure the network;**
- **tenants – the logical owners of the virtual network, who provide configuration and policy information through network apps; and**
- **management modules (MMs) – which are responsible for device administration.**

As illustrated in *Figure 1,* the SDN architecture comprises four planes: the data plane, the control plane, the application and the management plane. The data plane carries user traffic through the different NEs, which are dynamically programmed to respond to the policies of the different tenants. Forwarding policies are elaborated, and sent on by the control plane to each NE. The management plane is dedicated to infrastructure management, physical device management as well as platform management issues such as firmware and software upgrades[3, 4]. The application plane is constituted by all applications that program the network through interactions with the SDNC. These applications may be independent and owned by different tenants.

Networks that are built according to SDN architecture principles need to protect a number of key security assets:

- **availability – the network should remain operational even under attack;**
- **performance – the network should be able to guarantee a baseline bandwidth and latency in the event of an attack;**
- **integrity and confidentiality – control plane and data plane integrity and isolation should be upheld between tenants.**

To assure protection of these assets, a number of processes need to be in place:

### Authentication and authorization
Only authenticated and authorized actors should be able to access SDN components. The granularity of authentication and authorization must be detailed enough to limit the consequences of stolen credentials or identity hijacking.

### Resiliency
Networks must be able to recover as autonomously as possible from an attack, or a software or hardware failure. Alternatively, networks must be able to dynamically work around any affected functionality.

### Contractual compliance
To fulfill SLAs, mitigation techniques must be implemented, and proof that such techniques have been activated effectively must be provided.

### Multi-domain isolation
Systems must be able to isolate tenants in multiple domains, such as the resource and traffic domains.

The following forms of isolation apply:

- **resource isolation – prevents tenants from stealing resources, like bandwidth, from each other, and is required for SLA fulfillment; and**
- **traffic isolation – required by multi-tenant deployments, so a tenant can see its own traffic only (this requirement applies to both data plane and control plane traffic).**

### Repudiation
All actions carried out by all system actors – both internal and external – must be logged, and the all logs need to be secured.

### Transparency
Systems should provide visibility into operations and network status so they can determine the most appropriate action when issues arise. An active approach to security requires correct identification and classification of an issue so the most appropriate action to mitigate it may be chosen. Any action should be verified to ensure that it has been enforced effectively.

The potential vulnerabilities of SDN architecture are illustrated in Figure 2, which for the sake of simplicity shows only a subset of the possible major attacks.

## What's different about SDN security?
Many of the security issues related to SDN networks are similar to those that appear in traditional

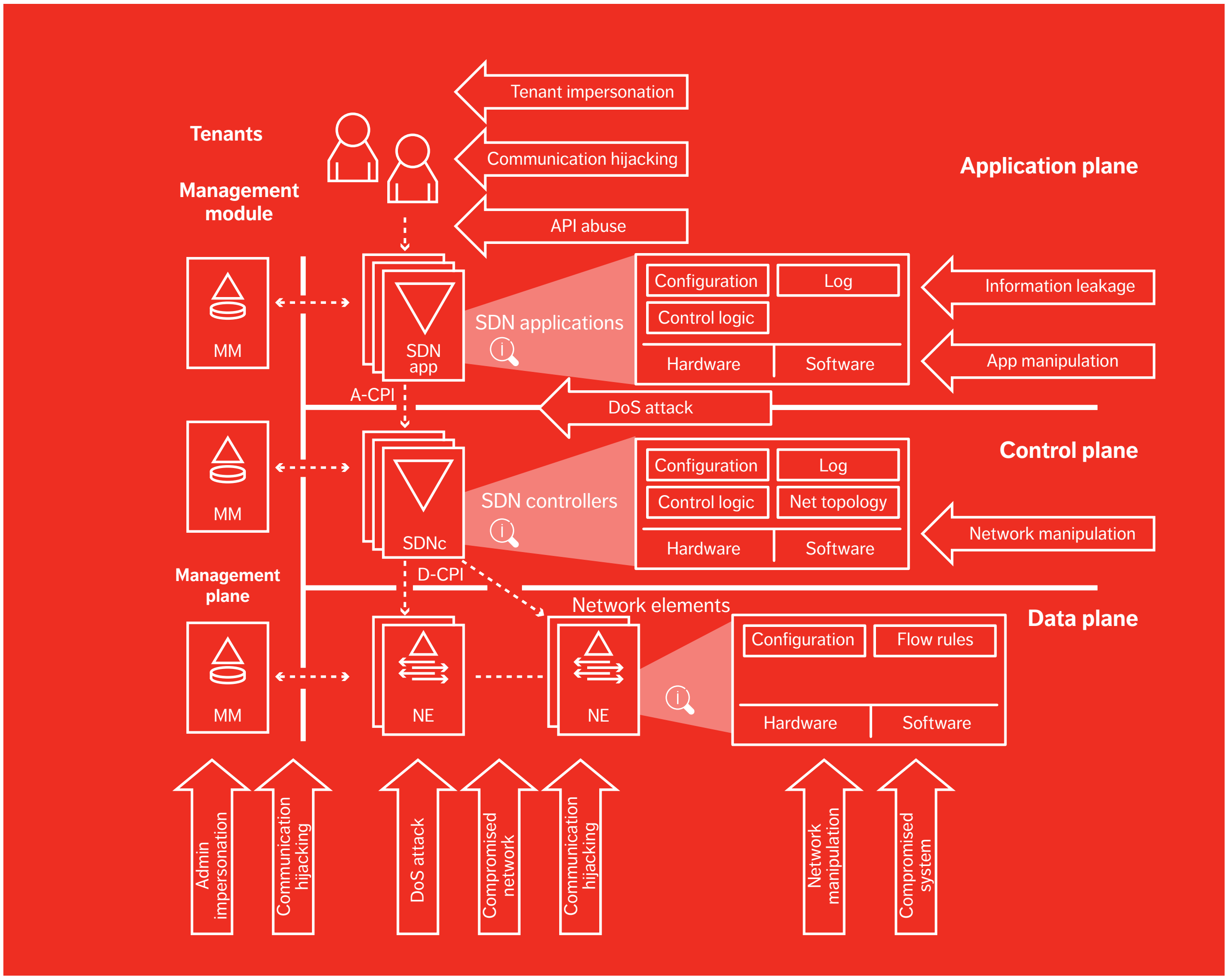


*Figure 2*
Potential vulnerabilities of SDN architecture

networks. What's interesting, however, is what sets SDN apart from traditional networks.

Compared with traditional networks, the separation of the control and data planes enables multi-tenancy and programmability, and introduces centralized management into the network architecture. In this new model, tenants run SDN apps that interface with the SDNC, which sends instructions to NEs. From a security perspective, the ability to share and dynamically operate the same physical network is one of the key security-related differences between SDN and traditional architectures. As such, SDN security issues relate to the new control plane model, and more specifically to securing inter-component communication, and controlling the scope of applications and tenants through specific APIs and access policies.

While it may sound like there are a number of obstacles to overcome, the programmability and centralized management brought about by SDN enables a much greater a level of autonomy to mitigate any security breaches – outweighing the need for additional technology.

### Centralized network management

In traditional networks, NEs tend to be monitored and managed individually. However, without the existence of standard protocols capable of interacting with all NEs irrespective of their vendor or generation, network management has become cumbersome. The SDN approach enables coordinated monitoring and management of forwarding policies among distributed NEs, resulting in a more flexible management process.

While there is a risk of the SDN control plane becoming a bottleneck, the fact that it has an overview of the entire network, makes it capable of mitigating any reported incident dynamically. For example, a DDOS attack can be detected and quickly mitigated by isolating the suspect traffic, networks or hosts. Unlike traditional DDOS appliances – which generally carry only a local view of the network – centralized elements possess a much broader view of network topology and performance, making the SDN an ideal candidate for the dynamic enforcement of a coherent security posture.

However, while it is clear that centralization provides significant benefits, it also presents a number of challenges, like the fact that the SDNC is a highly attractive attack surface. Thankfully, resiliency, authentication, and authorization address this risk, reducing the impact of attack.

### Resilient control plane

The three main elements of SDN are: SDN apps, the SDNC, and NEs. Given that control of the network is centralized, all communication within the control plane needs to be treated as critical, as an outage resulting from a successful attack may lead to an undesired impact on business continuity. If, for example, the SDNC is prevented from taking critical action to mitigate a DOS attack, the entire network and all of its tenants may be affected. To avoid this, the control plane needs a greater level of resiliency built into it.

To communicate with tenant applications and NEs, the SDNC exposes a set of interfaces. All these interfaces may experience heavy traffic loads, depending on the type and number of running applications. Traffic on the interfaces can be further impacted by NEs, for example, forwarding packets for which they have no forwarding rules. So, in terms of dependence on the SDNC, traditional networks appear to be more robust.

An effective way to improve the resilience of the centralized control plane and prevent the spread of DDOS control-plane attacks to the rest of the network is to rate-limit NEs in terms of bandwidth and resource consumption – such as CPU load, memory usage, and API calls.

Resilience can be further enhanced through proper resource dedication – where the SDNC authenticates each resource request, and subsequently checks requests against strong authorization control policies.

### Strong authentication and authorization

Authentication and authorization are the processes used to identify an unknown source and then determine its access privileges. Implemented correctly, these processes can protect networks from certain types of attack, such as:

- **provision of false (statistical) feedback to the system – for example, fooling the system into believing it is under attack, resulting in unnecessary deployment of countermeasures, which consumes resources and inevitably leads to suboptimal usage;**
- **modification of a valid on-path request – which results in a direct attack that alters network behavior;**
- **forwarding traffic that is not meant to be forwarded, or not forwarding traffic that should be – subverting network isolation; and**
- **gaining control access to any component – rendering the entire network untrustworthy.**

The critical nature of the SDNC dictates that additional security measures need to be taken to protect it. At the very least, traffic must be integrity protected to prevent tampering of on-path traffic, but even this level of protection does not secure control data.

Encryption is one way of preventing control data from being leaked. But, even together with integrity protection, encryption is not sufficient to protect against man-in-the-middle-type attacks. And so, all communication within the control plane must be mutually authenticated. Security protocols like TLS and IPSEC provide a means for mutual authentication as well as for replay attack protection, confidentiality, and integrity protection.

Mutual authentication does, however, present some difficulties, such as how to bootstrap security into the system. One way to solve this is by using security certificates. How then these certificates are issued, installed, stored, and revoked then becomes the significant security difficulty. Encryption and integrity protection without mutual authentication are less useful from a security point of view.

The problem with mutual authentication is that it requires previous knowledge of the remote communicating endpoint – unless a commonly trusted third party exists.

On a small scale, mutual authentication can be implemented manually – requiring administrators to install proper certificates or shared secrets on all endpoints. However, for complex and physically separated systems – and especially in networks where many SDN components can be created dynamically and administered by multiple parties – manual implementation may not be feasible.

The SDNC provides network configuration information through API calls to its services, which enables tenants to use SDN applications to control network behavior. This situation is somewhat alarming, given that physical hardware resources may be shared among rival tenants. While ordinary security measures – such as argument sanitization and validation – must be in place, the SDNC also needs a solid authentication, authorization and accountability infrastructure to protect the network from unauthorized changes. Strong authentication and authorization provides additional protection, as it prevents an attacker from impersonating an SDN component, especially the SDNC.

By enforcing strict authorization and accountability processes, damages can be limited, and reliable traces for forensics provided. Role-based access control (RBAC) is a commonly used approach for restricting the actions permitted by an application by assigning a role to it. Roles can be defined on a host, user or application basis.

In effect, RBAC is a security policy enforcing system. The fewer the number of permitted actions, the more limited the exploitable functionality. When implemented correctly, RBAC can be invaluable. Unfortunately, this approach is rather cumbersome in systems with very narrowly defined roles where frequent changes take place. At the other end of the scale, RBAC loses its edge if roles are too loosely defined.

For the purposes of system integrity assurance, every event that occurs in the system should be recorded in a log. How these logs are stored and secured against improper access also needs to be considered, and an external host is recommended.

## Multi-tenancy

Where networks are built using SDN techniques, it is possible for the same physical network to be shared among several tenants, which can in turn manage their own virtual networks. Multi-tenancy allows for better utilization of network resources, lowering the total cost of ownership. For tenants, SDN shortens the time taken to react to changing situations through, for example, automatic scaling of resources. To maintain an acceptable level of security, tenants should not be able to interfere with each other's networks, and need not even be aware that they are sharing network resources with others.

Tenant isolation (the separation of one tenant's resources and actions from another) is an important feature of SDN framework security.

## Control plane isolation

Isolation is one way to prevent the actions of one tenant from impacting others. This is a critical business aspect that must be strongly enforced. Tenant isolation is orchestrated by the SDNC, and implemented in SDN NEs through specific forwarding rules. While the burden of providing secure isolation lies with the SDNC, tenants also play an important role in sharing that burden.

The network provides isolation primarily on the link layer. If a tenant has weak network security procedures, information disclosure may occur, resulting in a breach of isolation at higher layers. For example, a rogue SDN app with privileges that span beyond isolation borders may impact overall network security by steering traffic to a third party (information disclosure) by over- or under-billing (theft of service) or by dropping traffic (DOS). The centralized nature of the SDN control plane further accentuates the impact of such attacks. Consequently, the task of providing isolation cannot be entirely offloaded onto the SDN network.

## Data plane isolation

Tenants running a business on virtual networks built using SDN may be subject to the same kind of network-based attacks as in traditional networks. However, due to the shared networking infrastructure, the impact of such an attack may be divided among some or even all of these tenants. This is a new risk, which may have a commercial impact; nobody wants to open a business next to a known (or perceived) troublemaker or one that is prone to attack.

So, for the data plane, flows associated with each particular tenant must remain isolated at all times. Isolation may be performed logically through overlay networks and enforced within the NEs. For example, by tagging the ownership of traffic generated by each tenant, the traffic can be carried over a shared infrastructure – once it has been encapsulated (tagged). Tunnels tagged for a given tenant are then forwarded to the virtual network for that tenant. Many alternative (and complementary) techniques are available for this type of encapsulation, including GRE, MPLS and IPSEC.

“AS THE SDNC IS SO CRITICAL, ADDITIONAL SECURITY MEASURES ARE NEEDED TO PROTECT IT”

Tagging is one way to perform logical isolation, but IP addresses can also be used, removing the need for specific tagging techniques. Bearing in mind that separate network function instances are not required to service different tenants, some network functionality can be shared by tenants as long as isolation is preserved and enforced.

In addition to logical isolation, traffic may be encrypted with specific tenant keys. This guarantees that in the case of logical encapsulation violation, the data traffic remains isolated and information cannot be leaked.

Isolation issues need to be resolved while bearing resource consumption in mind. While traffic isolation can help with data leakage, shared resource usage also requires resource isolation. For example, the existence of a forwarding loop within one tenant may potentially impact all tenants, as the problem overloads the underlying network equipment. To counteract this problem, the SDNC must enforce resource isolation, and use measures like rate limiting to minimize the impact that a tenant can have on the network.

## Programmability

One of the significant benefits brought about through SDN is programmability: the ability to configure a network efficiently, securely, and in a timely manner. SDN programmability exists in varying degrees of complexity and abstraction. At one end of the scale, programmability enables NEs to be dynamically reprogrammed to forward data flows according to their capabilities and higher-level policies in the network. At the other end, SDN apps enable tenants to programmatically issue run-time requirements to the network. All requests are consolidated by the SDNC, which fulfills higher-level requests from the capabilities available at the lower levels. To make this task trickier, SDN apps may issue orthogonal (mutually exclusive/contradicting) requests. The automated solution may then need to dynamically reconfigure a chunk of the SDN network – and all of this must happen within seconds or less.

The primary benefit that programmability brings for networks built using the SDN architecture approach is flexible control. The ability to control a network and apply changes in a timely manner increases the network's level of agility. Such flexibility can make the network more secure, as it is constantly monitored and designed to mitigate malicious behavior in more or less real time. The downside of the flexibility provided by programmability is the significant impact it has on security.

## Configuration coherency

Allowing tenants to issue programmatic changes to the network enables networks to adapt to changing conditions – increasing network agility. In practical terms, programmability can, for example, reduce the time it takes to set up a customer collaboration network from days or months to minutes or hours.

Programmability may also remove the need for manual configuration, which is prone to error. The result: the automatic reconfiguration of networks is feasible, providing the SDNC with a global view of the network, enabling it to perform sanity checking and regression testing so that new networks can be rapidly deployed.

Unfortunately, the flexibility provided by programmability allows tenants to make changes to the shared environment, which can cripple the operation of the entire network – either intentionally or unintentionally as a result of misinformation.

Ensuring coherency among the actions of the various SDN apps on the network also needs to be considered from a security point of view (as described in[5]). Consider the case where security and load-balancing applications are instantiated for a given tenant. A coherency conflict arises, for example, when the security application decides to quarantine a server, while the load-balancing application simultaneously decides to route traffic to the quarantined server – because it appears to have low load. To avoid coherency issues, the SDNC must be able to assess and eliminate the possible side effects of the acceptable network changes by each tenant, and to feature effective conflict resolution heuristics.

Another type of conflict arises due to the complexity of virtual network topologies, and the difficulty of maintaining a coherent security policy across a network. Special care is required for traffic that needs to be forwarded to security appliances for monitoring purposes. As the traffic or parts of it can be routed over different paths, methods need to be put in place to ensure that all the traffic is covered. Consequently, monitoring is necessary on all paths. Similar issues arise in traditional networks, but the increased service velocity offered by SDN architecture may fuel this type of conflict.

## Dynamicity

The dynamic and reactive nature of networks built using the SDN approach opens up new possibilities for fighting network attacks. Automated network reconfigurations, forwarding to honeypots, and black hole routing are just some of the techniques that can be employed. Service chaining is yet another technique that utilizes SDN properties and can be used to screen for malicious payload and trigger mitigating actions.

A network built using SDN techniques can do lower-layer analysis based on parameters such as data rate, source, and packet size, while the tenant can provide higher-layer analysis based on protocols, transport ports, and payload fingerprints. Once suspicious behavior has been detected, the network can use its programmability features to analyze the situation in more detail or trigger mitigating actions.

However, while the feedback system provides some advantages in terms of security, it also presents some issues. The interaction between the data plane and the control plane breaks the fundamental

SDN concept: the separation of these two planes. This in turn makes the data plane a stepping stone for attacking the control plane. As with other feedback loops, this interaction, unless managed appropriately, may lead to an oscillating situation that will eventually make the network unstable.

## Conclusion

The beauty of SDN lies in its ability as a technology to make networks flexible, ensure efficient use of resources, and facilitate a much higher level of system autonomy. Like any nascent technology, SDN should be handled cautiously to avoid it becoming an attack vector. However, SDN opens up new possibilities for the implementation of improved security mechanisms in the network, offering broader visibility, programmability, as well as a centralized approach to network management. ✱

### References

1. **Open Networking Foundation, 2014, SDN Architecture Overview, available at:** *http://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN-ARCH-Overview-1.1-11112014.02.pdf*
2. **ACM, 2013, Proceedings,Towards secure and dependable software-defined networks, abstract available at:** *http://dl.acm.org/citation.cfm?id=2491199*
3. **Ericsson, 2013, Ericsson Review, Software-defined networking: the service provider perspective, available at:** *http://www.ericsson.com/news/130221-software-defined-networking-the-service-provider-perspective_244129229_c*
4. **OpenDaylight project, available at:** *http://www.opendaylight.org/*
5. **CSL, SRI International, 2015, Proceedings, Securing the Software-Defined Network, available at:** *http://www.csl.sri.com/users/porras/SE-Floodlight.pdf*

## THE AUTHORS

### Kristian Slavov

◆ Works at Ericsson Security Research in Jorvas, Finland. He

has a background in programming and a keen interest in security, with more than 10 years of experience in this field. He holds an M.Sc. in telecommunications software from Helsinki University of Technology. He is also an avid canoe polo player.

### Daniel Migault

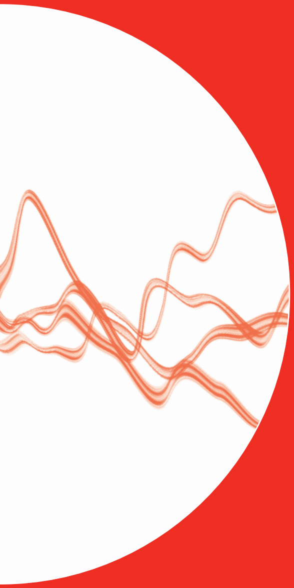
◆ Works at Ericsson Security Research in Montreal, Canada. He works on standardization at IETF and serves as a

liaison between IAB and ICANN/RSSAC. He used to work in the Security Department at Orange Labs for France Telecom R&D and holds a Ph.D. in Telecom and Security from Pierre and Marie Curie University (UPMC) and Institut National des Telecommunications (INT), France.

### Makan Pourzandi

◆ Works at Ericsson Security Research in Montreal, Canada. He has more than 15 years' experience in security for telecom systems, cloud, and distributed security and software

security. He holds a Ph.D. in parallel computing and distributed systems from the Université Claude Bernard Lyon 1, France, and an M.Sc. in parallel processing from École Normale Supérieure (ENS) de Lyon, France.



ISSN 0014-0171
284 23-3259 | Uen